Patrones con variables (I)

- Reglas: en la base de conocimiento

```
(defrule <nom-regla> [<comentario>]
         <elemento-codicional>*
         =>
         <acción>*)
```

  - Patrones:
    - Constantes
    - Variables
    - Restricciones
  - ...

- Variables:
  - Se declaran prefijando " ? " al nombre (**?x, ?sensor**)
  - No tienen tipo

Patrones con variables (II)

- Correspondencia de patrones con variables
  - Sea P un patrón con variables $V_1, \ldots, V_n$
  - Sea H un hecho en la Memoria de Trabajo (MT)

  P se corresponde con H si y sólo si es posible sustituir las variables $V_1, \ldots, V_n$ que aparecen en P de forma que el patrón constante P' que resulte coincide con H

  Ej:
```
(defrule informe
     (sensor ?id roto)
     =>
     (assert (cambiar ?id)))
```
```
(deffacts in
  (sensor s1 roto)
  (sensor s2 ok)
  (sensor s3 roto))
```

  - Cuando se produce una correspondencia, las variables $V_1, \ldots, V_n$ quedan ligadas a esos valores para el resto de la regla

Patrones con variables (III)

- El ámbito de la variable es el de la regla donde aparece
- Es un error encontrar una regla con una variable en la parte derecha que no haya sido ligada en la parte izquierda previamente
- Se generarán tantas activaciones de la regla como hechos satisfagan el patrón

  Ej:
```
(defrule informe
     (sensor ?id roto)
     =>
     (assert (cambiar ?id)))

(deffacts in
  (sensor s1 roto)
  (sensor s2 ok)
  (sensor s3 roto))
```

se activa la regla 2 veces

Patrones con variables
Ejemplo I

Cargar ejemp3.clp

```
(deftemplate persona
  (slot nom (type SYMBOL)) ;el nombre
(deftemplate hombre
  (slot nom (type SYMBOL)) ;el nombre
  (slot color-ojos (allowed-values azul gris verde turquesa marron)))
(deftemplate mujer
  (slot nom (type SYMBOL))
  (slot color-ojos (allowed-values azul gris verde turquesa marron)))
(deffacts hombres-y-mujeres
  (hombre (nom maurice)     (color-ojos azul))
  (hombre (nom jean-michel) (color-ojos marron))
  (mujer  (nom mary)        (color-ojos azul))
  (mujer  (nom sally)       (color-ojos verde))
  (hombre (nom olof)        (color-ojos turquesa))
  (hombre (nom gandalf)     (color-ojos azul))
  (mujer  (nom rosa)        (color-ojos verde)))
(defrule hombres  "los hombres son personas"
  (hombre (nom ?nombre)) => (assert (persona (nom ?nombre)))
(defrule mujeres  "las mujeres son personas"
  (mujer (nom ?nombre)) => (assert (persona (nom ?nombre)))
```

Patrones con variables (IV)

- Cuando una variable se liga, queda ligada para el resto de la regla, tanto en su parte izquierda (LHS) como en la derecha (RHS)

```
(defrule mismo-color-ojos
      "dos mujeres con ojos iguales"
      (mujer (nom ?m1) (color-ojos ?color))
      (mujer (nom ?m2) (color-ojos ?color))
      =>
      (assert (mismo-color-ojos ?m1 ?m2)))
```

  - Se añadirían los siguientes hechos:

```
(mismo-color-ojos rosa rosa)
(mismo-color-ojos rosa sally)
(mismo-color-ojos sally rosa)
(mismo-color-ojos sally sally)
(mismo-color-ojos mary mary)
```


Patrones con variables (V)

- Algunos usos incorrectos
  - Usar una variable por primera vez en un elemento condicional **not**:

```
(defrule regla-erronea-1
   (not (hombre (color-ojos ?color)))
   =>
   (assert (falta-color ?color)))
```

  - Usar variables que no vayan a ligarse:

```
(defrule regla-erronea-2
   (or (hombre (color-ojos ?color1))
       (mujer  (color-ojos ?color2)))
   =>
   (assert (uno-de-dos ?color1 ?color2)))
```

evaluación en cortocircuito

Patrones con variables
Ejemplo II

- Considerar el problema de calificación de los alumnos
  - 2 exámenes: **n1** y **n2**
  - Notas posibles: **susp | aprob | not | sobr**
  - Nota final: **apto** si ambas notas (**n1** y **n2**) no son **suspenso**
  - Pepe: aprobado (**n1**) y notable (**n2**)
  - Mari: aprobado (**n1**) y suspenso (**n2**)
  - Loli: notable (**n1**) y no presentado (**n2**)

Ej:

```
(defrule regla-apto
   (oav ?n n1 ~susp)
   (oav ?n n2 ~susp)
   =>
   (assert (oav ?n total apto)))
```

```
(deffacts notas
   (oav pepe n1 aprob)
   (oav pepe n2 not)
   (oav mari n1 aprob)
   (oav mari n2 susp)
   (oav loli n1 not))
```


Ejercicio-8

- Dadas las siguientes plantillas:

```
(deftemplate hombre
  (slot nom (type SYMBOL))
  (slot color-ojos (type SYMBOL)))
```

```
(deftemplate mujer
  (slot nom (type SYMBOL))
  (slot color-ojos (type SYMBOL)))
```

- ¿Es correcta la siguiente definición?

```
(defrule regla-misteriosa-1
   (initial-fact)
   (not (hombre (color-ojos ?color)))
   (mujer (color-ojos ?color))
   => (assert (color-femenino ?color)))
```

- ¿Es equivalente a la siguiente regla?

```
(defrule regla-misteriosa-2
   (initial-fact)
   (mujer (color-ojos ?color))
   (not (hombre (color-ojos ?color)))
   => (assert (color-femenino ?color)))
```

Funciones

- Usan notación prefija:

```
(nombre-funcion arg1 arg2 ... argn)
```

- Tipos:
  - Manejo de hechos y agenda

```
(assert ...), (retract ...) ...
```

  - Aritméticas (+, -, *, /)

```
(+ 3 5)    (* ( + 1 2) (+ 3 4))
```

  - Comparación numérica (>, <, >=, <=, =, <>)

```
(> (+ 1 2) (+ 0 3))
```

  - Comparación (**eq**, **neq**)

```
(eq hola hola)   (neq hola hola)
```


Elementos condicionales: test

- Además de los elementos condicionales vistos (**and**, **or**, **not**) tenemos el elemento condicional **test**:

```
(test <llamada-a-predicado>)
```

- Se satisface si y sólo si el resultado devuelto por la llamada al predicado es **TRUE**
- **test** es una palabra reservada del lenguaje
- Se puede usar para comprobar el valor de las variables que hayan sido ligadas en la correspondencia de patrones

Ej:

```
(deftemplate persona
  (slot nom)
  (slot edad)))

(defrule jubilado
  (persona (nom ?n) (edad ?e))
  (test (> ?e 64))
  =>
  (assert (jubilado ?n)))
```


Ejercicio-9

- A partir del ejemplo de las calificaciones:
  - Las notas pasan a ser numéricas
  - Pepe: **n1**=5,5 y **n2**=8
  - Mari: **n1**=6 y **n2**=3,5
  - Loli: **n1**=7
- Reescribir el programa para que calcule la nota final sabiendo que para ser apto hay que sacar al menos 5 puntos en cada una de las notas